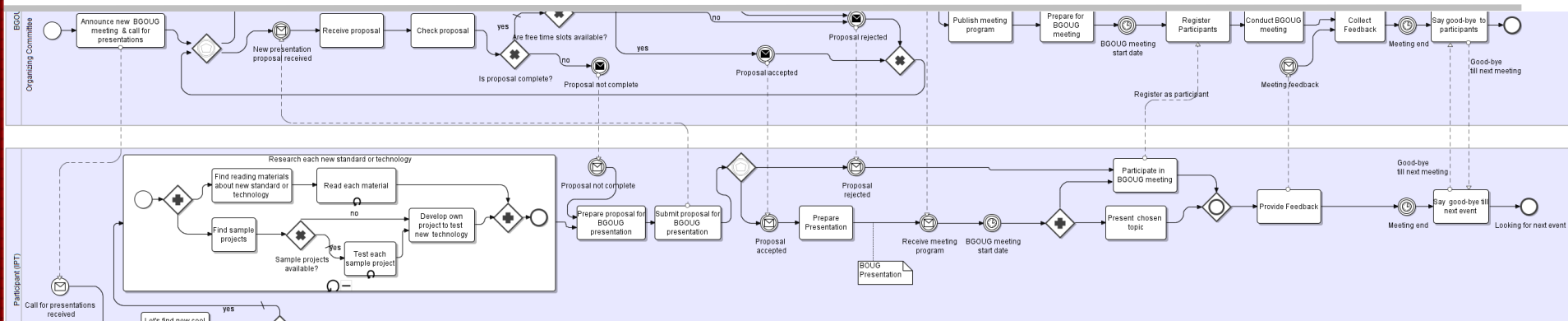


Технологии за генериране на динамични уеб страници. Използваемост, Web 2.0, RIA & AJAX. Java™ сървлети.



Траян Илиев

IPT – Intellectual Products & Technologies
e-mail: tiliev@iproduct.org
web: <http://www.iproduct.org>

Oracle®, Java™ and EJB™ are trademarks or registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners. Oracle®, Java™ и EJB™ са търговски марки на Oracle и/или неговите подразделения. Всички други търговски марки са собственост на техните притежатели.

Съдържание

1. Технологии за реализация на динамични уеб страници
2. Използваемост на уеб приложенията
3. Web 2.0 (социален уеб) и Rich Internet Applications (RIA)
4. Asynchronous JavaScript™ and XML (AJAX)
5. Server Push (Comet), HTML 5 SSE, WebSockets
6. Java™ Servlet™ технология
7. Предимства на сървлетите
8. Жизнен цикъл на сървлетите
9. Основни методи на класовете GenericServlet и HttpServlet
10. Конфигуриране на сървлетите в web.xml

WEB 2.0 и социални технологии

- Бъдещето на Web – три гледни точки:



Необходимост от динамични уеб страници

- Уеб страница, която използва:
 - данни въведени от клиента
 - корпоративни бази от данни
 - данни, които често се променят
 - електронни магазини и електронен бизнес
 - приложения, които позволяват на потребителите активно да допринасят за качеството на сайта, като добавят информация (коментари, рейтинги и други) към уеб сайта в стил уеб 2.0
 - уеб-базирани комуникационни приложения – форум, кратки съобщения, електронна поща

Технологии за динамични уеб страници

- От страна на сървъра:
 - CGI и Perl
 - Java Servlet (JSP, JSTL, JSF, Apache Struts, Apache Wicket, Apache Click, Play!, Spring, GWT, Vaadin, ...)
 - ASP.NET MVC, MonoRail, ...
 - Ruby on Rails, ...
 - PHP (Zend Framework, Symfony, ...)
- От страна на клиента (уеб браузър):
 - Flash & Flex (ActionScript, MXML)
 - JavaScript™ (ECMAScript)
- Комбинация от страна на клиента и сървъра:
 - AJAX + уеб услуги (SOAP, REST)

Цел: подобрена използваемост

- Минимален мрежови трафик
- Съгласуване с очакванията на потребителя - съгласувани метафори
 - вербални
 - интерфейсни
 - КОМПОЗИТНИ
- Без разсейващи елементи
- Достъпност за всички потребители
- Изтегляне само на необходимата информация
- Потребителят и неговите потребности в центъра на разработката

Феноменът Web 2.0

- Web 2.0 – Интернет като платформа за общуване, **създаване** и **споделяне** на съдържание – блогове, RSS/Atom, коментари, снимки, аудио, видео (**social media**)
- Колаборативно конструиране на колективно знание – **Participatory, Decentralized, Linked, Emergent**
- Нови ценности и нови възможности – мрежови ефект, постоянна еволюция, Taksonomy -> Folksonomy
- Отворени стандарти и свободен софтуер за създаване
- Service-Oriented Architectures (SOA)
- Разделяне на данни и презентация
- Богат, реагиращ потребителски интерфейс

Rich Internet Applications (RIA)

- Rich Internet Application (RIA) – уеб приложение, което характеристиките на традиционните десктоп приложения като:
 - Богат уеб интерфейс – drag-and-drop, анимирани ефекти, локални изчисления, богати компоненти – бутони, менюта, таб-панели, слайдери, индикатори на прогреса
 - Отговарящи интерактивни приложения
 - Баланс клиент-сървър
 - Асинхронна комуникация
 - Мрежова ефикасност
- Не изисква инсталация, осъвременяването е автоматично
- Достъпно е независимо от мястото и вида на устройството

Asynchronous JavaScript & XML - AJAX

- Ajax – един нов подход към веб приложенията, Джеси Гарет, февруари 2005
<http://www.adaptivepath.com/publications/essays/archives/000385.php>
- презентация базирана на стандарти HTML 5 или XHTML, CSS
- динамична визуализация и взаимодействие с използване на Document Object Model (DOM)
- обмяна и манипулация на данни чрез XML и XSLT или JavaScript Object Notation (JSON)
- асинхронно извл. на данни чрез XMLHttpRequest
- и JavaScript който обединява всичко в едно

АЈАХ и традиционните веб приложения

Основна разлика:

- Ајах приложенията са базирани на обработка на събития и данни
- Традиционните веб приложения са базирани на показване на страници и преходи между тях

Проблеми свързани с AJAX (1)

- Sandboxing
- Изключено скриптиране
- Скорост на обработка при клиента
- Време за изтегляне на скрипта
- Загуба на интегритет
- Проблеми с индексирането от машините за търсене
- Достъпност
- По-сложно разработване и тестване на приложенията
- По трудно измерване на времето за отговор – 2 цикъла

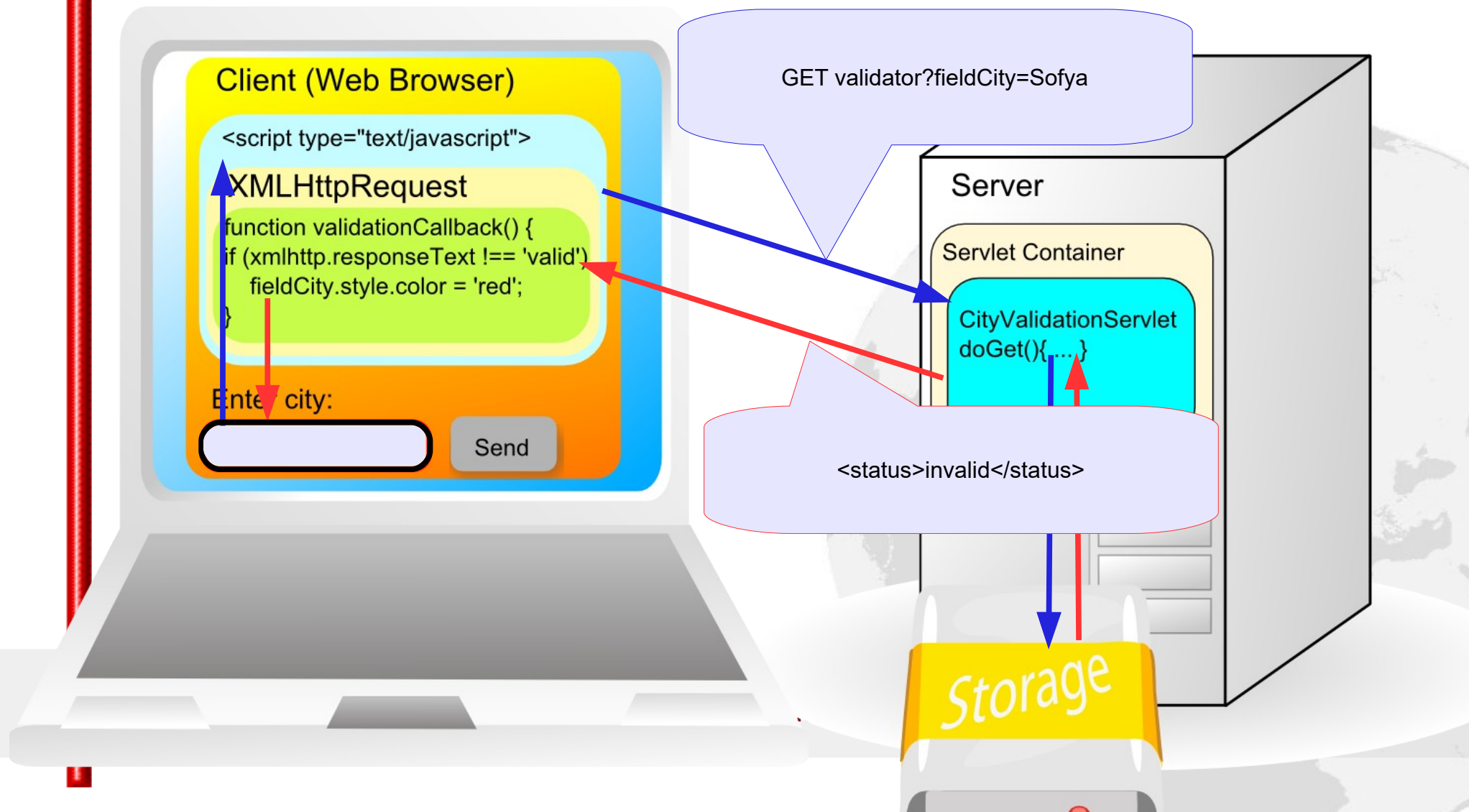
Проблеми свързани с AJAX (2)

- При операции, които изискват време е необходима визуална обратна връзка за статуса на операцията (индикатор на прогреса)
- Бавните операции могат да доведат до неочакван за потребителя **update** на страницата
- Преди **HTML 5** да осигури API за взаимодействие с **History** на брауъра не работеше бутона “**Back**”
- Преди **HTML 5** да осигури API за взаимодействие с брауъра беше по трудно да се **bookmark**-не конкретното състояние на страницата
- Доскоро **Cross Domain AJAX** заявките бяха проблем

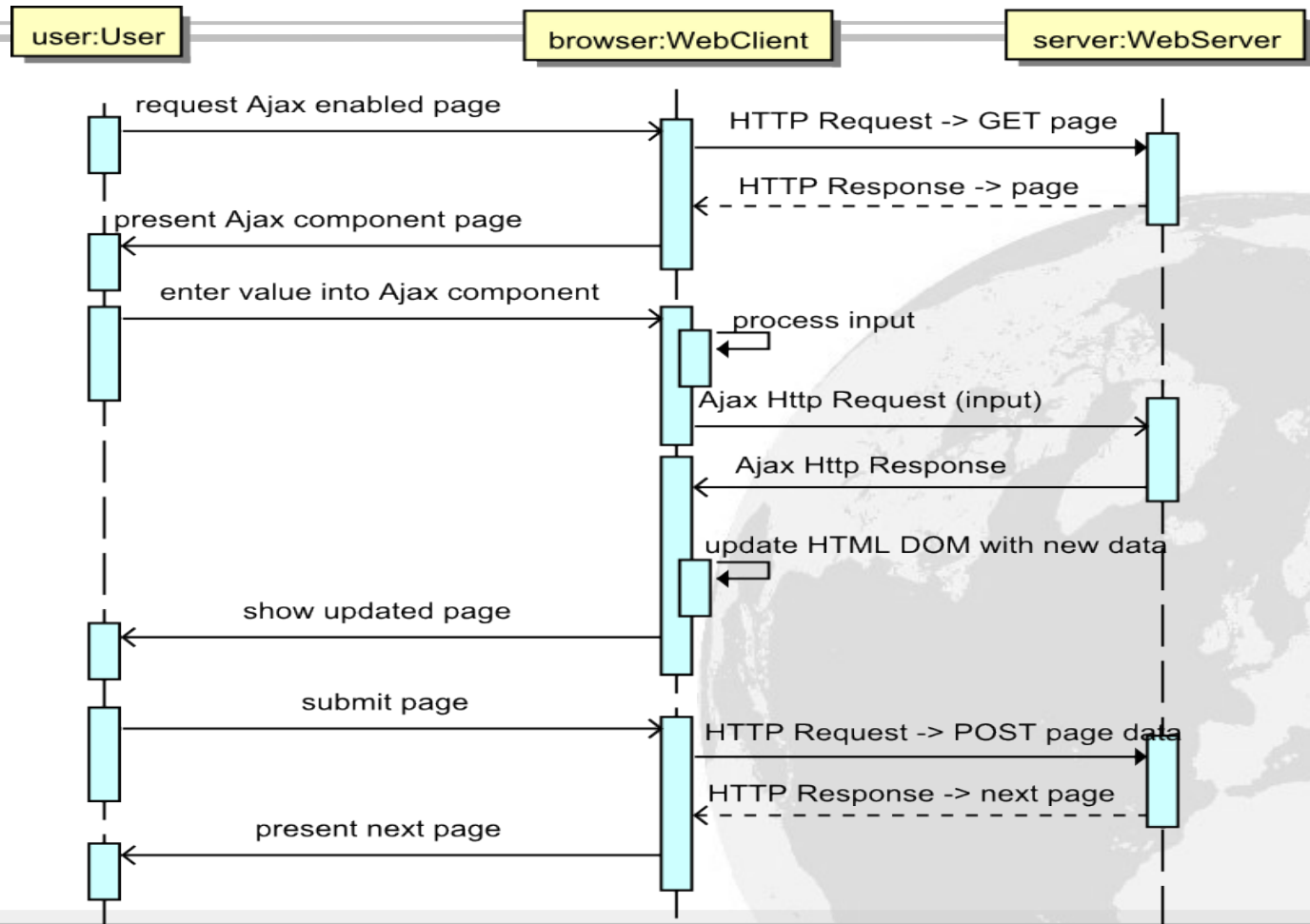
AJAX и еволюцията на уеб

- JavaScript
- Фреймове
- Скрити фреймове
- DHTML и DOM
- IFrames
- XMLHttpRequest

AJAX – механизъм на взаимодействие



AJAX – механизъм на взаимодействие



AJAX - технологии

- HTML 4/5 и XHTML
- JavaScript, DHTML и DOM
- CSS
- XML и XSLT
- XMLHttpRequest
- SOAP, WSDL, UDDI, REST
- JSON, JSONP, ...


Примери за AJAX приложения

- Google Suggest
- Gmail
- Google Maps
- Google Docs
- Google Calender
- Product Search on Amazon – A9
- Blogger
- Yahoo! News
- и много други

Базова структура за обработка на синхронна AJAX заявка

```
var method = "GET";  
var url = "resources/ajax_info.html";  
  
if (window.XMLHttpRequest) { // IE7+, Firefox, Safari, Chrome, Opera,  
    xmlhttp=new XMLHttpRequest();  
} else { // IE5, IE6  
    xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");  
}  
  
xmlhttp.open(method, url, false);  
xmlhttp.send();  
document.getElementById("results").innerHTML =  
    xmlhttp.responseText;
```

isAsynchronous = false



Обработка на AJAX заявка за получаване на XML документ с аутентификация

```
if (window.XMLHttpRequest) { // IE7+, Firefox, Safari, Chrome, Opera,
    xmlhttp=new XMLHttpRequest();
} else { // IE5, IE6
    xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
}
xmlhttp.open("GET", "protected/product_catalog.xml", false,
    "trayan", "mypass");
xmlhttp.send();
if (xmlhttp.status == 200 &&
    xmlhttp.getResponseHeader("Content-Type") == "text/xml") {
    var xmlDoc = xmlhttp.responseXML;
    showBookCatalog(xmlDoc); // Do something with xml document
}
```

Обработка на **синхронна** AJAX заявка за получаване на XML документ

```
function showBookCatalog(xmlDoc){
    txt("<table><tr><th>Title</th><th>Artist</th></tr>");
    var x=xmlDoc.getElementsByTagName("TITLE");
    var y=xmlDoc.getElementsByTagName("AUTHOR");
    for (i=0;i<x.length;i++) {
        txt=txt + "<tr><td>"
            + x[i].firstChild.nodeValue
            + "</td><td>" + y[i].firstChild.nodeValue
            + "</td></tr>";
    }
    txt += "</table>"
    document.getElementById("book_results").innerHTML=txt;
}
```


Базова структура за обработка на асинхронна AJAX заявка

```
if (window.XMLHttpRequest) { // IE7+, Firefox, Safari, Chrome, Opera,
    xmlhttp=new XMLHttpRequest();
} else { // IE5, IE6
    xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
}
xmlhttp.onreadystatechange = function() {
    if (xmlhttp.readyState==4 && xmlhttp.status==200){
        callback(xmlhttp);
    }
}
xmlhttp.open(method, url, true);
xmlhttp.setRequestHeader("Content-type","application/x-www-form-
    urlencoded");
xmlhttp.send(paramStr);
```

Callback function

isAsynchronous = true

XMLHttpRequest.readyState състояния

Код	Значение
1	след като XMLHttpRequest.open() е извикан успешно
2	заглавните части на отговора на HTTP заявката (HTTP response headers) са успешно получени
3	начало на зреждане на съдържанието на HTTP отговора (HTTP response content)
4	съдържанието на HTTP отговора е заредено успешно от браузъра

AJAX заявки независими от браузъра

```
function getXMLHTTP() {  
    var xmlhttp = null;  
    if (typeof XMLHttpRequest != "undefined") {  
        xmlhttp = new XMLHttpRequest();  
    } else {  
        try {  
            xmlhttp = new ActiveXObject("Msxml2.XMLHTTP");  
        } catch (e) { }  
        if (xmlhttp == null) {  
            try {  
                xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");  
            } catch (e) { }  
        }  
    }  
    return(xmlhttp);  
}
```

Основни подходи за Ајах разработка

- Направи си сам
- Клиентска JavaScript библиотека: jQuery, YUI, DojoToolkit
- Използване на JavaScript UI библиотека – Facebook React, Polymer, Web Components

[<https://smthngsmwhr.wordpress.com/2015/04/13/web-components-and-friends-react-angular-polymer/>]

- Клиентски JavaScript / TypeScript MVC/MV* фреймуърк: Backbone.js, Knockout.js, Ember.js, AngularJS / Angular 2 и др. [<http://codebrief.com/2012/01/the-top-10-javascript-mvc-frameworks-reviewed/>]
- Client-Server фреймуърк - JSF, PrimeFaces и др.
- All in one подход – Google Web Toolkit

Server Push (Comet)

- Дълготрайна HTTP заявка, която позволява на уеб сървъра да праща данни (server push) към браузъра, без клиентът (браузърът) да ги е поискал явно
- **Publish/subscribe** модел с множество канали
- Примерни приложения: **synchronous conferencing, instant messaging, email, market data distribution (stock tickers), online chat/messaging systems, auctions, online betting and gaming, sport results, monitoring consoles, sensor network monitoring**
- Реализации:
 - Streaming – Hidden iframe, XMLHttpRequest
 - Ajax with long polling – XMLHttpRequest polling, Script tag polling
 - **HTML 5 Server Side Events (SSE), HTML 5 WebSocket, HTTP/2, Java applets, Flash XMLSocket relays, BOSH (XMPP)**

Предимства на сървлетите (1)

- **Ефикасни** – няма нужда от отделна инстанция на сървлета за всяка заявка
- **Лесни за използване** – има готови методи за основните задачи (работа с HTTP хедъри, кукита, проследяване на сесии)
- **Мощни** – специална поддръжка и възможности за комуникация с уеб контейнера (транслиране на пътища)
- **Платформено независими** – могат да се прехвърлят без промяна към друг сървър
- **Безплатни уеб сървъри** и Java servlet и JSP контейнери

Предимства на сървлетите (2)

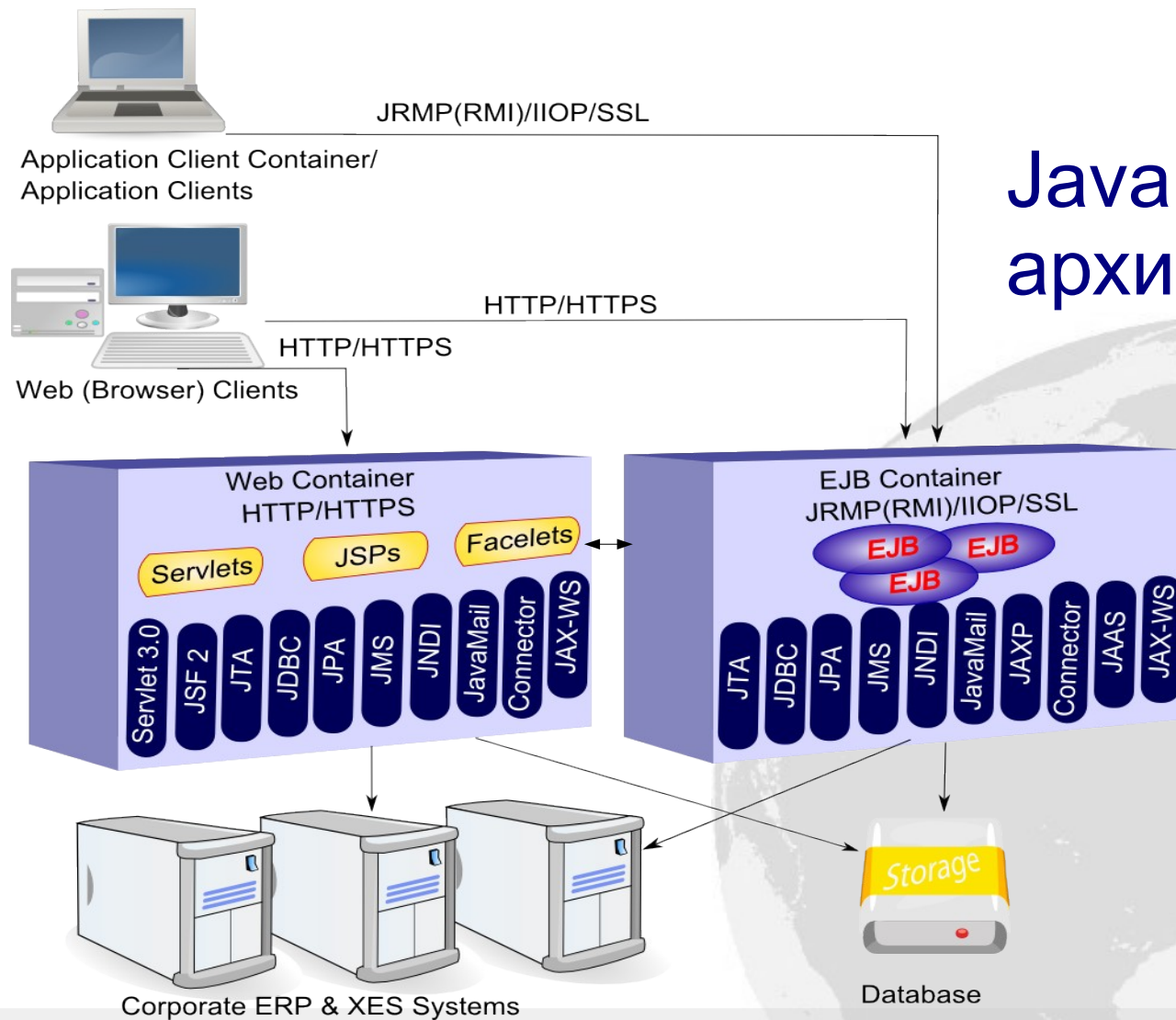
- **Сигурни и надеждни** – вградени механизми за сигурност на езика Java + декларативна сигурност чрез деплоймънт дескриптора
- **100% ОБЕКТНО ОРИЕНТИРАНИ**
- **Широка индустриална поддръжка** от най-големите разработчици на софтуер:
 - Apache, Oracle, IBM, Hewlett-Packard, Inria, Novell, Red Hat, SAP, Sybase, Caucho, Sun/iPlanet, New Atlanta, ATG, Fujitsu, NEC, TmaxSoft, Lutris, Silverstream, World Wide Web Consortium (W3C) ...
 - Plugins за IIS и Zeus

Използване на сървлетите

- Платформи:
 - Windows, Unix/Linux, MacOS, Solaris и др.
- Използват го авиолинии, компании за електронна търговия, хотели, финансови институции и др.
- **Водеща технология за изграждане на средни и големи уеб приложения и корпоративни портали**

Сървърна поддръжка

- Java™ EE 6/7 сървъри:
 - GlassFish (<https://glassfish.dev.java.net>)
 - RedHat's JBoss (<http://www.jboss.org>)
 - Apache Geronimo (<http://geronimo.apache.org/>)
 - Apache TomEE и TomEE+ (<http://tomee.apache.org/apache-tomee.html>)
 - Oracle's WebLogic Suite – part of Fusion Middleware
 - IBM's WebSphere
 - SAP Netweaver
 - Resin, JOnAS, JEUS, . . .
- Олекотени веб сървъри: Apache Tomcat, Jetty и много други



Java™ EE 6 архитектура

Жизнен цикъл на сървлета

- Зареждане на класа на сървлета от веб контейнера
- Веб контейнерът създава инстанция на класа
- Инициализира инстанцията на сървлета като извиква метода **init()**
- Извиква **service()** метода за всяка заявка подавайки **request** и **response** обекти като аргументи
- Преди да деактивира (премахне) сървлета контейнера извиква неговия метод **destroy()**

Основна структура на сървлети (1)

- Основни методи на класа HttpServlet:
 - doGet – за HTTP GET заявки
 - doPost – за HTTP POST заявки
 - doPut – за HTTP PUT заявки
 - delete – за HTTP DELETE заявки
 - init and destroy – за управление на ресурсите
 - getServletInfo – дава информация за сървлета
 - service – получава всички HTTP заявки и играе ролята на диспечер – **не трябва да се предефинира директно**

Основна структура на сървлети (2)

- Основни методи на класа **GenericServlet**:
 - **getInitParameter**, **getInitParameterNames** – дават възможност за декларативно конфигуриране
 - **getServletConfig** – връща обект от тип **ServletConfig**, който съдържа информация предавана от уеб контейнера на сървлета
 - **getServletContext** – връща обект от тип **ServletContext** дефиниращ множество методи, които сървлетът използва за да комуникира с контейнера – например да получи MIME типа на файл, да диспечеризира заявки или да пише в log файл

Клас HttpServlet – методи: doGet, doPost

- `response.setContentType("text/html");`
- `PrintWriter out = response.getWriter();`
- `out.println(docType + "<HTML>\n" +
"<HEAD><TITLE>Hello</TITLE></HEAD>\n" + "<BODY
BGCOLOR=\"#FDF5E6\">\n" + "<H1>Hello</H1>\n" +
"</BODY></HTML>");`

Инсталиране на сървлети

- Инсталиране и конфигуриране на сървлет и JSP™ софтуер
- Динамична регистрация на сървлети – анотации `@WebServlet` и `@WebInitParam`.
- Уеб компоненти и WAR архиви
- Структура на дескриптора на разпространението (deployment descriptor) – `web.xml`

Пример за сървлет с използване на @WebServlet и @WebInitParam анотации (1)

```
@WebServlet(urlPatterns = "/HelloWorld",
    initParams = {
        @WebInitParam(name="bgcolor", value="yellow"),
        @WebInitParam(name="message", value="Hello from Servlet 3.0")
    })
public class HelloWorld extends HttpServlet {
    private String bgcolor;
    private String message;

    public void init() throws ServletException {
        bgcolor = getInitParameter("bgcolor");
        bgcolor = (bgcolor != null) ? bgcolor: "white";
        message = getInitParameter("message");
        message = (message != null) ? message: "Hello";
    }
}
```

Пример за сървлет с използване на @WebServlet и @WebInitParam анотации (2)

```
protected void doGet(HttpServletRequest request,
    HttpServletResponse response) throws ServletException,
    IOException {
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    out.println("<html>");
    out.println("<head>");
    out.println("<title>Hello World!</title>");
    out.println("</head>");
    out.println("<body bgcolor='" + bgcolor + "'>");
    out.println("<h1>" + message + "</h1>");
    out.println("</body>");
    out.println("</html>");
}
```

Дескриптор на разпространение web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">
  <servlet>
    <servlet-name>DispatcherServlet</servlet-name>
    <servlet-class>invoicing.DispatcherServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>DispatcherServlet</servlet-name>
    <url-pattern>/DispatcherServlet</url-pattern>
  </servlet-mapping>
```


Дескриптор на разпространение web.xml (2)

```
<session-config>
  <session-timeout>
    30
  </session-timeout>
</session-config>
<welcome-file-list>
  <welcome-file>index.jsp</welcome-file>
</welcome-file-list>
</web-app>
```

Структура на дескриптора на разпространение web.xml

- icon
- display-name
- description
- distributable
- context-param
- filter
- filter-mapping
- listener
- servlet
- servlet-mapping
- session-config
- mime-mapping
- welcome-file-list
- error-page
- taglib
- resource-env-ref
- resource-ref
- security-constraint
- login-config
- security-role
- env-entry
- ejb-ref
- ejb-local-ref

Обработка на параметри на форми с помощта на сървлети.

Методи на **javax.servlet.ServletRequest** за обработка на параметри на форми:

- **getParameter(String name)** – връща стойността на параметър с даденото име във формата като низ (String)
- **getParameterValues(java.lang.String name)** – връща всички стойности на параметъра като масив от низове
- **getParameterNames()** - връща **java.util.Enumeration<String>** с имената на всички параметри във формата
- **getParameterMap()** - връща асоциативен списък (**java.util.Map<String, String[]>**) с всички имена и стойности на параметри във формата

Отстраняване на грешки (debugging) на сървлети

Методи за откриване и отстраняване на грешки в сървлети:

- **System.out.println()** и **javax.servlet.GenericServlet.log()** – отпечатване на междинни резултати за диагностика на работата на сървлета в **log** файла на сървъра
- Използване на инструментите за отстраняване на грешки (**Debugger tools**) на интегрираната среда за разработка (**IDE**) – например **Eclipse** и **NetBeans** предлагат такива
- Използване на **Firebug** и други подобни инструменти за инспекция на **HTML** и **JavaScript** кода, който се визуализира и изпълнява вътре в уеб браузъра, както и за преглед на съдържанието на **HTTP** заявките и отговорите от сървъра
- Използване на **отделни класове** за отделните задачи

Ресурси

- Hall, M., Brown, L., Core Servlets and JavaServer Pages, Prentice Hall, 2004 – <http://pdf.coreservlets.com/>
- Java EE 7 Tutorial – <https://docs.oracle.com/javaee/7/tutorial/>
- Flanagan, D., JavaScript: The Definitive Guide, 5th Edition. O'Reilly, 2006.
- Гарет, Д., Ајах-един нов подход към веб приложенията, 2005
<http://www.adaptivepath.com/publications/essays/archives/000385.php>
- W3Schools AJAX Tutorial-<http://w3schools.com/ajax/default.asp>

Благодаря Ви за Вниманието!

Въпроси?